REMARKS

Claims 1-8, 10 and 11 were rejected under 35 U.S.C. 102(b) as being anticipated by Pekowski (U.S. patent No. 5,946,486, hereinafter "Pekowski01"). Claims 9, 12, 13, 16 and 17 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce (U.S. patent No. 6,880,149, hereinafter "Cronce"), while claims 14 and 15 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce as applied to claim 12 above, and further in view of Downs et al. (U.S. patent No. 6,226,618, hereinafter "Downs"). Claims 18 and 21-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce and Pekowski02 (U.S. Patent 6,003,095, hereinafter "Pekowski02"). Claim 24 is rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Pekowski02. No new matter has been introduced in this reply to the outstanding office action. Claims 3, 4 and 15 have been cancelled. Claims 1, 5, 6, and 12 have been amended.

SPECIFICATION

An amendment to paragraph [0066] in view of the Examiner's constructive comments have been made. As amended, it is believed that the objection to the specification has been overcome.

CLAIM OBJECTIONS

Claim 21 has been amended in view of the Examiner's comment, as amended it is believed that the objection to claim 21 has been overcome.

REJECTIONS UNDER 35 U.S.C. § 102(b)

Claims 1-8, 10 and 11 were rejected under 35 U.S.C. 102(b) as being anticipated by Pekowski01. The cited Pekowski01 reference teaches using a shadow dynamic link library (DLL) which intercepts calls from a calling application program executable. As shown and discussed in reference to FIG. 3, the shadow dynamic link library is named the same as the original target DLL, and the target DLL is renamed to not conflict. The shadow DLL is then used to trace events occurring upon entry and exit from a software module without having to modify the software module. The FIG. 6 in Pekowski01 diagram shows the calling application 700, the shadow DLL 725 and the target DLL 710.

Claim 1 has been amended to recite in part (additions in bold):

"using a third software module having one or more stubs ***that comprise code segments that are callable by the first software module as an intermediary, the one or more stubs*** for performing said first method, the one or more stubs being used to enter the second software module and identify said functionality, ***said call being made according to a first calling convention, said third software module using a second calling convention different from said first calling convention to invoke said functionality in the second software module, the second calling convention comprising a non-standard calling convention that preserves a return address across more than one call boundary***;

verifying that the call originated from a source that is permitted to invoke said functionality, the one or more stubs ***from the third software module*** comprising data required during said verification ***by the second software module, said data required during said verification being mixed into instruction streams provided by the one or more stubs, the data also comprising information that is used to identify a function that will be invoked after the verification***;

performing said ***functionality that includes sensitive functions at the second software module, the sensitive functions comprising verifying and authenticating the call from the first software module***; and

returning ***from the second software module*** to said first software module ***that issued the call and bypassing said third software module and the one or more stubs***."

Support for this amendment can be found for example in paragraph [0046] to [0050]. It is believed that, as amended, claim 1 has been clarified in order to overcome the noted rejection in view of the cited Pekowski01 reference. A comparison of FIG. 6 in Pekowski01 and FIG. 5 of the current application shows some of the major differences between the two, with the newly introduced limitations further clarifying the main differences. For example, claim 1 now recites in part "using a third software module having one or more stubs ***that comprise code segments that are callable by the first software module as an intermediary***".

Pekowski01 fails to teach such a module having one or more stubs that comprise such code segments. Furthermore, as shown in FIG. 6 of the present application and as now claimed in claim 1, the method of claim 1 includes "returning to said first software module ***that issued the call and bypassing said third software module and the one or more stubs".*** In one embodiment, a BBJump causes a jump from the blackbox shown in FIG. 5 to the client application bypassing the blackbox stubs (see FIG. 6 of the current application). This is something that is neither taught nor suggested in the cited Pekowski01 reference. If the calling application 700 is construed to be the first software module in claim 1, the shadow DLL 725 is construed to be the third software module, and the target DLL 710 is construed to be the second software module (black box in FIG. 5) as is done in the analysis presented in the Office Action, then the return has to occur between the target DLL and the calling application without going back through the shadow DLL. This is not the case in Pekowski01 since, as noted in the current Office Action, it is the common_exit 755 within the shadow DLL 725 itself that makes the return to the calling application. In the present application, the blackbox (second software module) is bypassed completely. As mentioned in the Office Action "each call to the target DLL and return from the target DLL passes through the shadow DLL", which is clearly not the case as recited in claim 1.

Pekowski02 has been introduced as attempting to establish the jumping past an intermediate entity for efficiency purposes as noted in page 19 of the Office Action. It does not seem that, in Pekowski01, there is any suggestion or need to use the technique taught by Pekowski02. Since Pekowski01 is concerned with generating a shadow DLL that intercepts calls from a calling application (see FIG. 6 and its accompanying discussion), the target DLL 710 in Perkowski 01 needs to return back through the shadow DLL 725 and not bypass it. Attempting to combine Pekowski02 with Pekowski01 would simply take away the shadow DLL functionality that Pekowki01 is teaching. As such, it seems that this is not a proper combination.

Another difference between currently amended claim 1 and the cited references is that claim 1 recites "verifying that the call originated from a source that is permitted to invoke said functionality, the one or more stubs ***from the third software module*** comprising data required during said verification ***by the second software module, said data required during said verification being mixed into instruction streams provided by the one or more stubs,***

the data also comprising information that is used to identify a function that will be invoked after the verification". This is much different than that which is described in Col. 10, lines 27-30 of Pekowski01 which fails to teach including information that is used to identify a function that will be invoked after verification.

Another difference between claim 1 and Pekowski01 is that claim 1 recites "performing said ***functionality that includes sensitive functions at the second software module, the sensitive functions comprising verifying and authenticating the call from the first software module***". The second software module which is called by the first software module in claim 1 is much different than the call application 700 and target DLL 710 described in FIG. 6 of Pekowski01. As recited, the second software module (see blackbox 136 in FIG. 5 of the current application) performs sensitive functions that include verifying and authenticating the call generated by the calling application. In Pekowski01, the target DLL 710 simply does not perform any of these functions. The other limitations added to claim 1 add further features which are neither taught or suggest in Pekowski01, nor by attempting to combine the other cited references. In view of the above, it is believed that claim 1 and dependent claims 2, 5-11 which add further nonobvious features to claim 1, are in condition for allowance.

REJECTIONS UNDER 35 U.S.C. § 103(a)

Claims 9, 12, 13, 16 and 17 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce, while claims 14 and 15 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce as applied to claim 12 above, and further in view of Downs. Claims 18 and 21-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Cronce and Pekowski02. Claim 24 is rejected under 35 U.S.C. 103(a) as being unpatentable over Pekowski01 in view of Pekowski02.

Independent claim 12 has been amended to recite in part that:

the first program module being called by the second program module via a third program module having one or more stubs with code segments that are callable by the

second program module as an intermediary, the one or more stubs comprising data required during a verification by the first software module, said data required during said verification being mixed into instruction streams provided by the one or more stubs, the data also comprising information that is used to identify a function that will be invoked after the verification;

based on the result of said determining act, permitting execution of said first program module to proceed ***and returning to said second software module which issued the call and bypassing said third software module and the one or more stubs,***

wherein said first program module comprises cryptographic functionality that stores and obscures a decryption key and that uses said decryption key to decrypt content.

Claim 12 has been amended in somewhat similar manner to claim 1 (although it is worth noting that the first software module and second software module in claim 12 are reversed as to that which is claimed in claim 1). As such, claim 12 and dependent claims 13, 14, 16 and 17 which add further nonobvious features to claim 12 are also believed to be in condition for allowance.

Independent claim 18 has been amended to further recite in part:

"wherein said function is not exposed to said calling entity, and wherein said function is exposed to an intermediate entity that is callable by said calling entity, said intermediate entity calling upon the program module to perform said function on behalf of said calling entity***, said intermediate entity comprising one or more stubs that comprise data required by the logic to verify the identity of the calling entity, the data being mixed into instruction streams provided by the one or more stubs, the data also comprising information that is used to identify the function***." As mentioned previously above, this is something that the cited references including Pekowski01 fail to teach or suggest. As such claim 18 and dependent claims 21-23 which add further nonobvious features are also believed to be in condition for allowance.

With regard to independent claim 24, Applicants respectfully traverse the rejection in view of the previously made comment that Pekowski01 discloses returning through the shadow DLL 725 (see FIG. 6) and the 1-to-n exit_functions 750-770 and the common-exit routine 755 (which track the 1-to-n entry points). The Perkowski02 "jump past" the demand load code would simply render Pekowski01's shadow DLL functionality inoperable, not more efficient as mentioned in the Office Action. Thus, Applicants believe the attempted combination of Pekowski01 and Pekowski02 in this manner is improper. Given the above, it is believed that claim 24 is in condition for allowance.

CONCLUSION

In view of the above amendments and remarks, applicant respectfully submits that the present invention is in condition for allowance. Reconsideration of the application is respectfully requested.

Date: May 5, 2008

/Kenneth R. Eiferman/
Kenneth R. Eiferman
Registration No. 51,647

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439